

Course for developing Full Stack Java Project

In every project we perform
2 kinds of operations

1. Development
2. After development
[build, deploy, test, ...]

Every project has 3 parts

1. UI
2. Programming
3. Data Base

Full Java Development
courses are divided into
5 modules

1. Java Module
2. Data Base Module
3. UI Module
4. DevSecOps tools
5. AWS basics

Courses for Project development

Java Module
1. Core Java
2. Adv Java
3. Spring
4. SpringBoot
5. Microservices

Data Base
Oracle
1. SQL
2. PL/SQL

UI Module
1. HTML
2. CSS
3. Java Script
4. Bootstrap
5. React JS

Courses for after development operations [building, deploying, testing, ...]

DevSecOps Tools
GIT, GitHub
Maven, Gradle
Log4J, SLF4J
Junit, Mockito, Jmeter, JIRA,
Jenkins, Docker, Kafka,
etc...

AWS Basics
1. Installing web server
2. Installing Data base
3. Deploying project
4. Running project
5. Providing project access to client

Java Module

1. Core Java (HK)
2. Adv Java
3. Spring Boot
4. Microservices
5. Mini Projects

Data Base Module

- Oracle
- 1. SQL
- 2. PL/SQL

UI Module

1. HTML
2. CSS
3. Java Script
4. Bootstrap
5. React JS

DevSecOps Module(15+ tools)

1. Maven 2. Gradle
3. Log4J 4. SLF4J
5. Junit 6. Mockito 7. HttpUnit
8. Git/GitHub
9. Jira 10. Jenkins 11. Docker
12. Kafka , ...

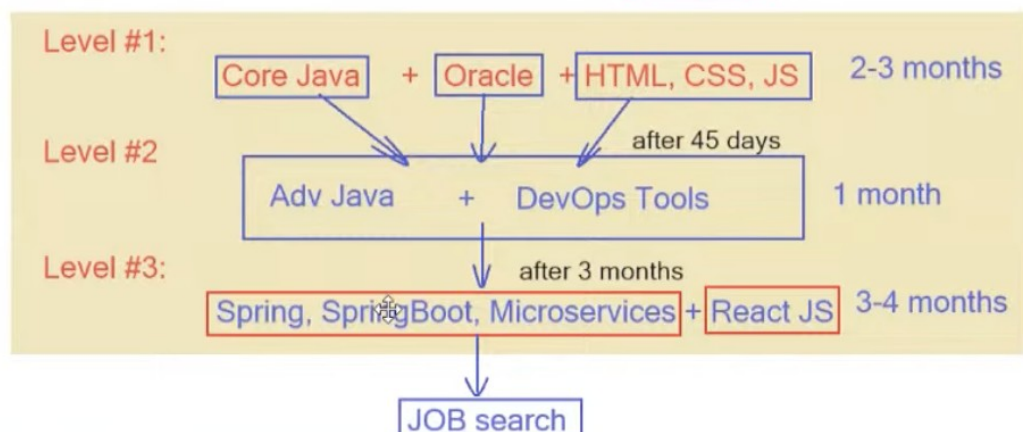
AWS Basics Module

1. Creating account in AWS
2. Installing Webserver(Tomcat)
3. Installing Database (Oracle)
4. Deploying project
5. Providing access to client

Full Stack Java Learning Road Map

We can learn Full Stack Java courses in 2 modes **Duration:**

1. Sequentially - one course after another course (1+ yrs)
2. In parallel - per day [2 or 3] courses (6+ months)



1. Every project is divided into three parts or layers I

1. UI (User Interface)

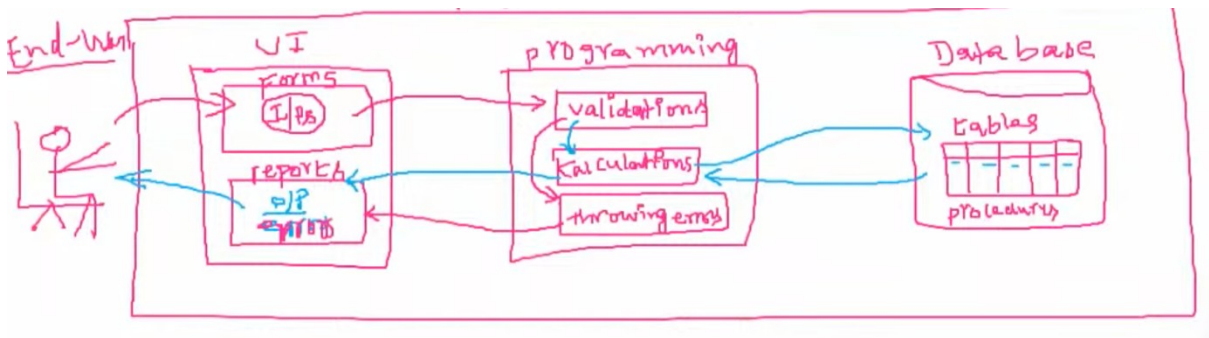
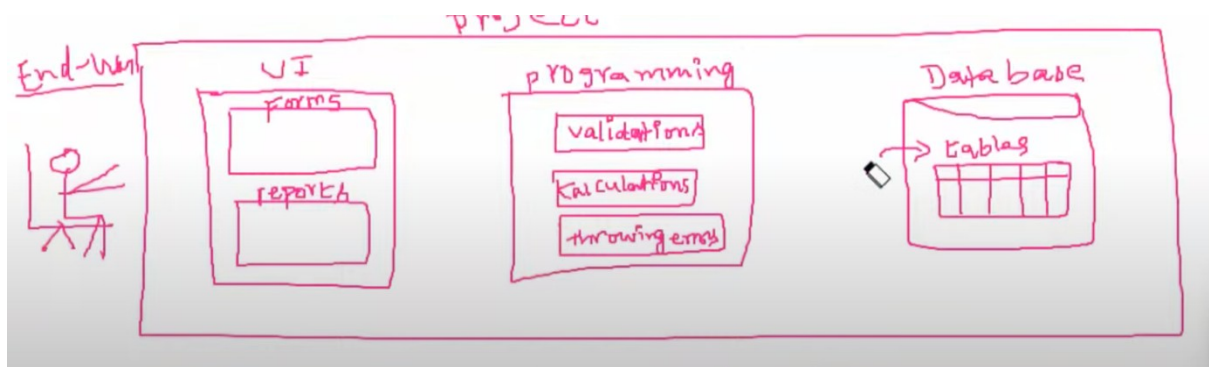
2. Programming

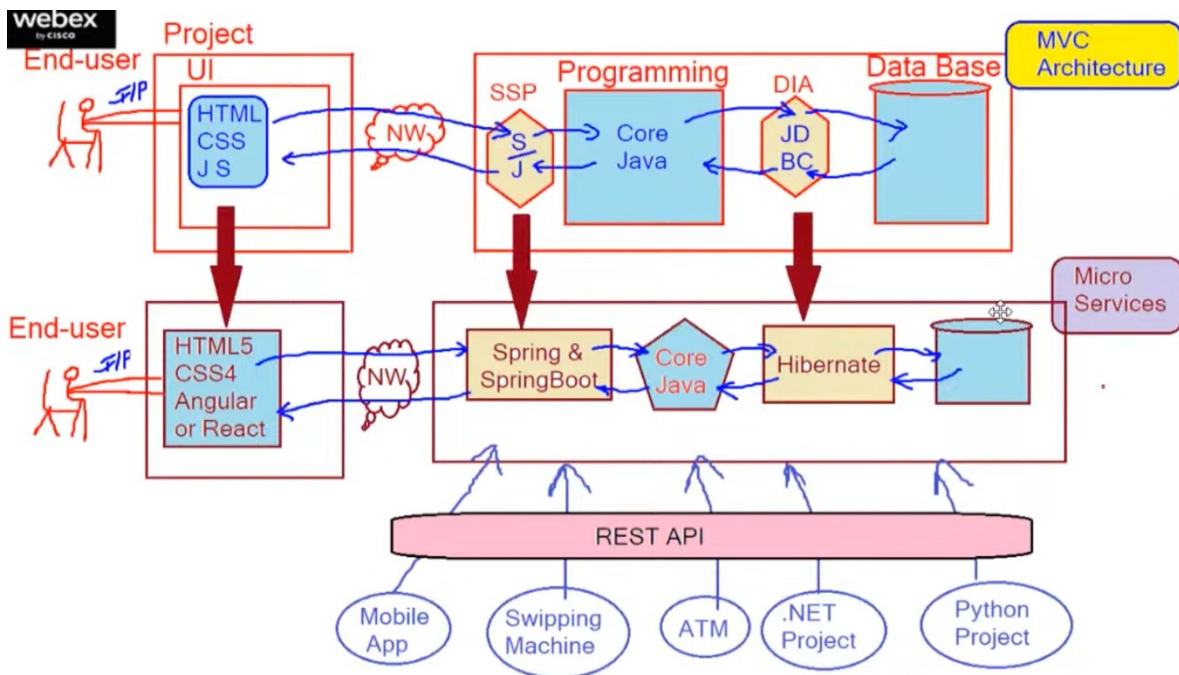
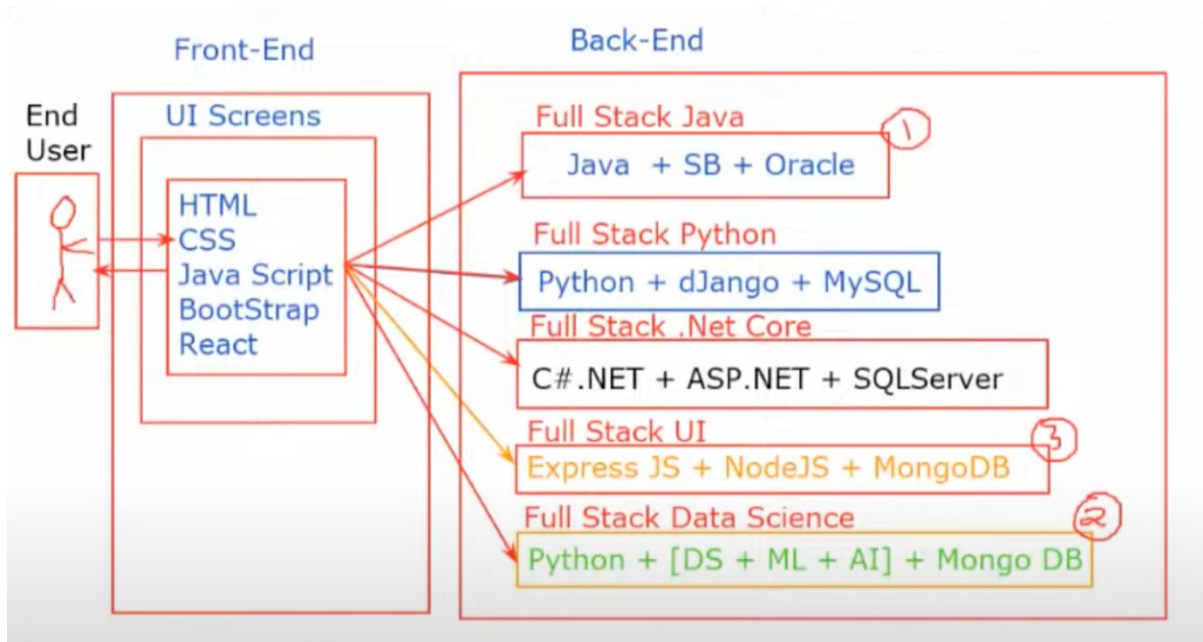
3. Data Base

1. HTML, CSS, Java Script, BS, React JS

2. CJ, AJ, Spring, SB, MS

3. Data Base Oracle(SQL, PL/SQL)





Old = mvc arch

New = micro services

1. What is JAVA?

ANS: Java is a programming language and an object-oriented language.

Platform independent language.

Types of applications?

- All available softwares or projects or applications are broadly divided into 2 types

1 Stand-alone applications

2 Internet applications

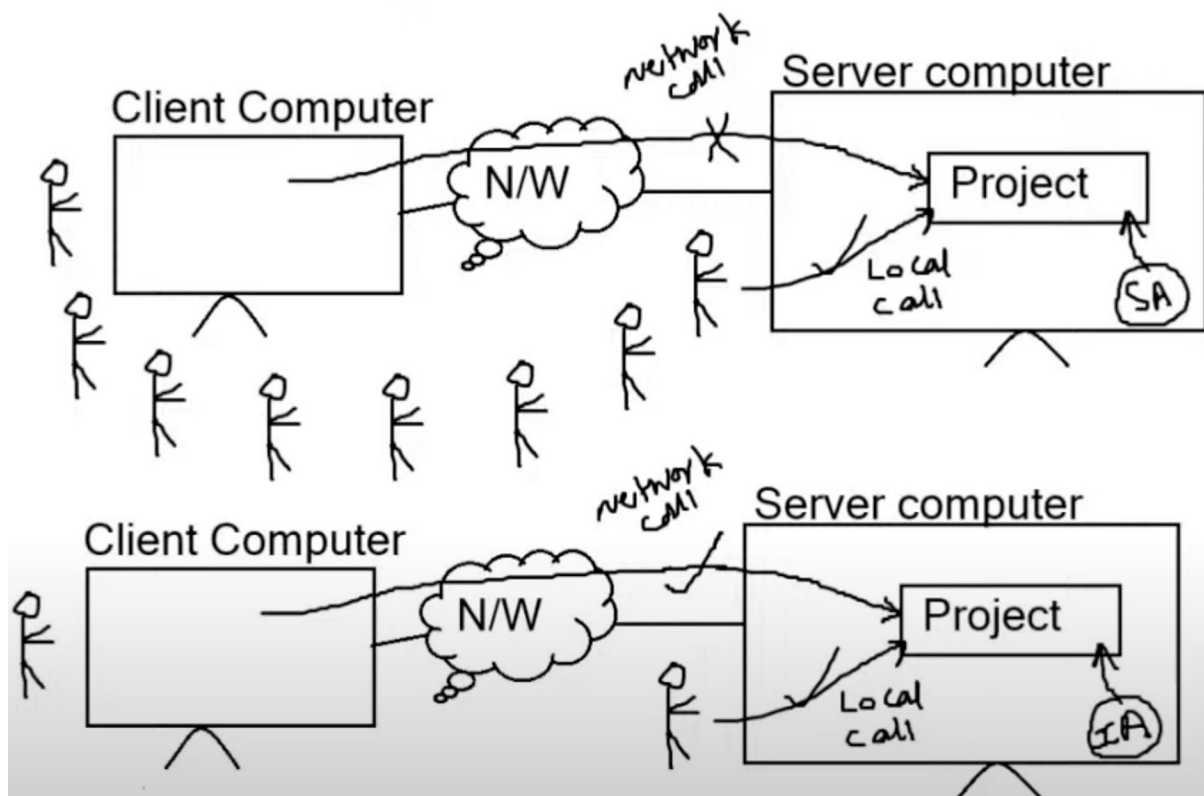
- *An application that is installed in one computer and can be accessible only within this computer and can't be accessible from other computers via network is called stand-alone application*

For example: Calculator, MS-Office, Antivirus, VLC, ... are Stand Alone.

- *An application that is installed in one computer and can be accessible from other computers throughout the world via network is called internet application*

For example:

Amazon, irctc, instagam, bookmyshow, youtube, gamil, Naresh IT, HKProgrammingWorld, etc... are Internet Applications.



What is a Platform, Platform dependency, Platform independency?

Platform Dependency -

A program that is developed in one OS, ex: windows, if it can not execute in a different OS,
ex: Linux or Mac.

We call this program as platform dependent program
The programming language using which we develop this program is called platform dependent programming language.

Platform Independency -

A program that is developed in one OS, ex: windows, if it can be executed in different OS,
ex: Linux or Mac,

We call this program as platform independent program
The programming language using which we develop this program is called platform independent programming language.

If a program or a software is directly executed by OS, then this program or software is a platform dependent program and platform dependent software

If a program is executed by intermediate software, like JVM, PVM, CLR, then this program is platform independent program

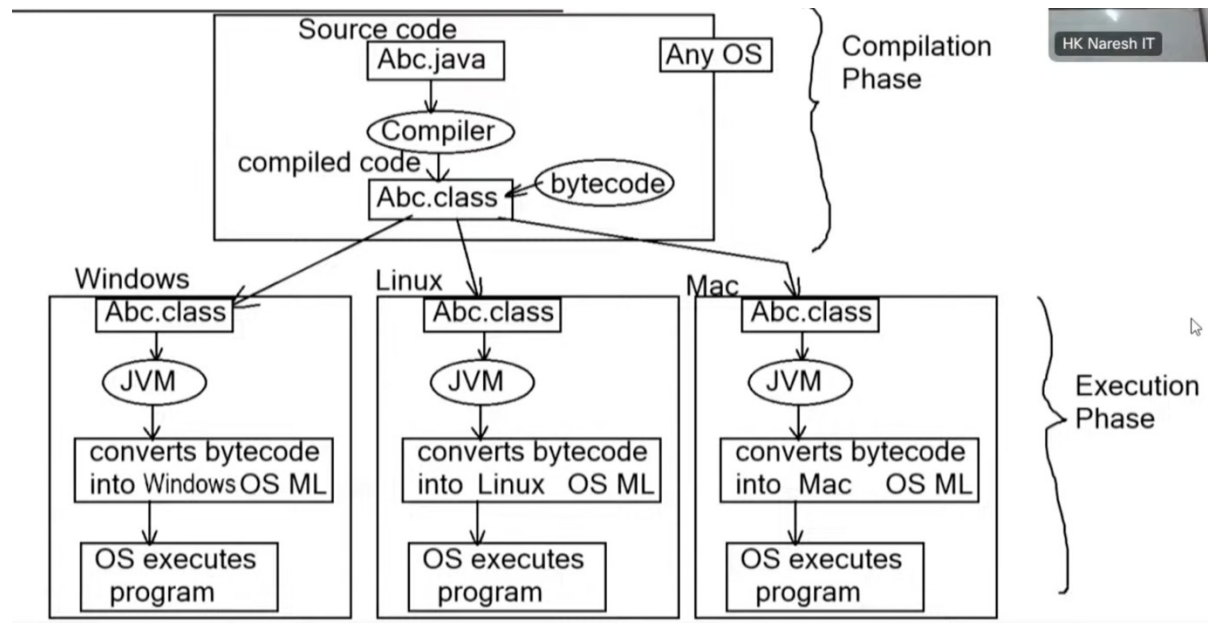
Why C and C++ PLs are platform dependent?

ANS:

The C and C++ programs are compiled targeting to one particular OS.
Hence C and C++ programs compiled code contains Machine Language of a particular OS whose format is not understandable to different OS and can not be executed by different OS.
so C and C++ programs are called platform dependent.

How java achieved platform independency?

Mainly because of byte code & JVM.



In Java, program is not compiled targetting to any particular os
Java program is compiled into bytecode, which is an intermediate language all OS

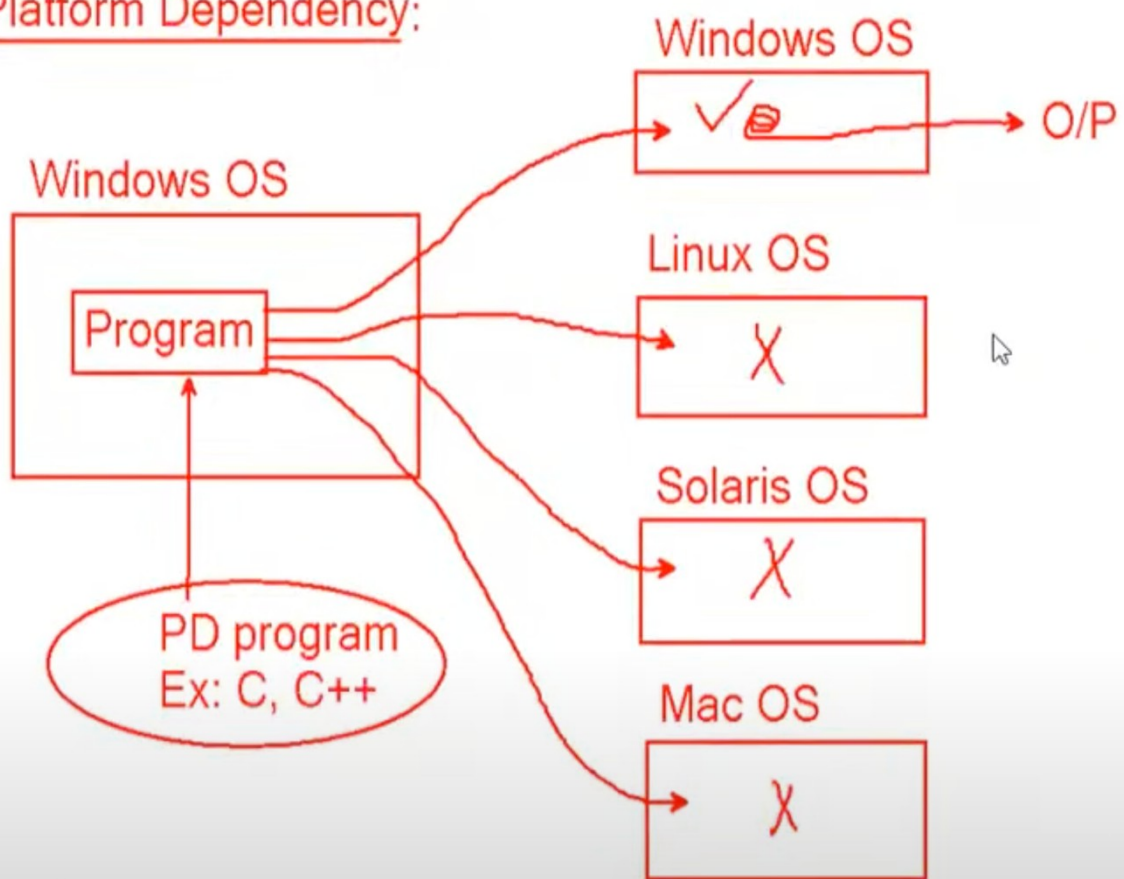
Then later this byte code is executec by JVM by translated it into the current OS's Machine Language code.

Because we have separate JVM for every OS, hence the same compiled code,

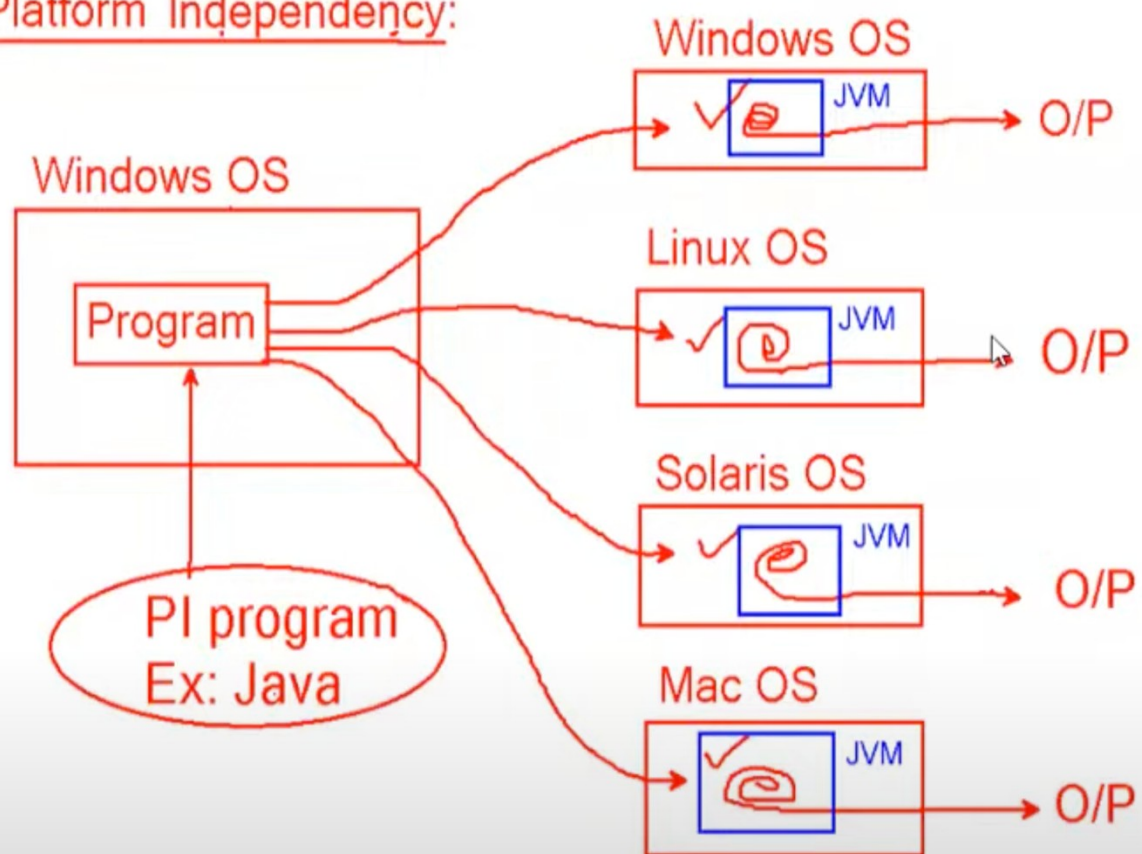
byte code is executed in all OS with the help of JVM

- Hence Java achieved Platform Independency, that is WORA(Write Once Run Anywhere).

Platform Dependency:



Platform Independency:



Java program development, compilation and execution procedure

- In the Java programming language, all source code is first written in plain text files ending with the *“.java”* extension.
- Those source files are then compiled into *“.class”* files by the *javac* compiler.
 - A *“.class”* file does not contain code that is native to your processor; it instead contains *bytecodes* — the native language of the Java Virtual Machine.
- The *java* launcher tool then runs your application with an instance of the Java Virtual Machine.

Q: What does a compiler do?

A: A compiler translates source code written in a high-level programming language into machine code (or an intermediate form like bytecode) so that it can be executed by a computer. In Java, the compiler converts the source code (.java file) into bytecode and saves it as a .class file, which is then executed by the Java Virtual Machine (JVM).

***JAVA SOFTWARE = JDK + JRE**

JDK = COMPILER + JRE

JRE = JVM + API”

Introduction

Java is an object-oriented, class-based, secured, platform-independent, and general-purpose programming language. Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystem's Java platform. Java programming language is based on the write once, run anywhere (WORA) principle, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java Virtual Machine (JVM) regardless of the underlying operating system.

Different Editions of Java

There are four editions of the Java programming language.

- **Java Platform, Standard Edition (Java SE)**
 - **Java Platform, Enterprise Edition (Java EE)**
 - **Java Platform, Micro Edition (Java ME)**
 - **JavaFX**
1. **Java SE: Java SE, also known as Core Java, is the most basic and standard version of java. It consists of a wide variety of general-purpose APIs (like java.lang, java.util) as well as many special-purpose APIs. Java SE is used to create Desktop applications. It defines everything from the basic types and objects of java programming language to high-level classes that are used for networking, security, database access, graphical user interface development (GUI), and XML parsing.**
 2. **Java EE: The Java EE platform is built on top of the Java SE platform. The Enterprise Edition version of java has a much larger usage of Java, like the development of web services, networking, server-side scripting, and other various web-based applications. Java EE uses HTML, CSS, JavaScript, etc., so as to create web pages and web services. It is also one of the most widely used web development standards.**

- 3. Java ME:** The Java ME platform is widely used for developing embedded systems, mobiles, and small devices. Java ME uses many libraries and APIs of Java SE, as well as many of its own. The basic aim of this edition is to work on mobiles, wireless devices, set top boxes, etc. Most of the apps developed for the phones were built on Java ME only.
- 4. JavaFX:** JavaFX is another edition of java technology, which is now merged with Java SE 8. It is mainly used to create rich GUI (Graphical User Interface) in java apps. It is supported by both desktop environments as well as web browsers.

Features and Uses of Java

The features of the Java programming language are given below.

- 1. Simple:** Java programming language is easy to learn and its syntax is simple and easy to understand. According to Sun Microsystem, the java programming language is simple because the java syntax is similar to C++. So, it is easier for the programmer to learn java after C++. Java has removed many rarely-used features like explicit pointers, operator overloading, etc. In java, there is no need to remove unused objects because there is an automatic garbage collection in java.
- 2. Object-Oriented:** Java is an object-oriented programming language. Everything in java is an object. We can use the functionality of our program by using objects. Object-oriented programming language simplifies software development and maintenance by providing some rules. The basic concepts of OOPs are classes and objects, packages, polymorphism, inheritance, abstraction, encapsulation, and etc.
- 3. Platform independent:** Java is a platform-independent programming language. We can write java code once and run it anywhere. Java code can be run on multiple platforms, for example, Windows, Linux, Mac, etc. Java code is compiled by the compiler and converted into byte code. This byte code is platform-independent code because it can be

run on multiple platforms.

- 4. Robust:** Java programming language is robust because it uses strong memory management. There is an automatic garbage collection in java to destroy the objects which are unused. Exception handling is another feature that makes java programming robust.
- 5. Portable:** Java programming language is portable because it enables you to carry byte code to any platform.
- 6. Multithreaded:** Java also supports multithreaded programming. We can create multiple threads in java. The main advantage of multithreading is that it doesn't occupy the memory for each thread. It shares a common memory area. Threads are important for web applications, multimedia applications, etc.
- 7. Distributed:** Java programming language is distributed because it facilitates the user to create distributed applications. We can create distributed applications in java by using RMI and EJB.
- 8. Secure:** When a java program is compiled, it generates a byte code that is in the non-readable form. The java byte code cannot be read by humans. That makes the java language secure.

Uses of Java

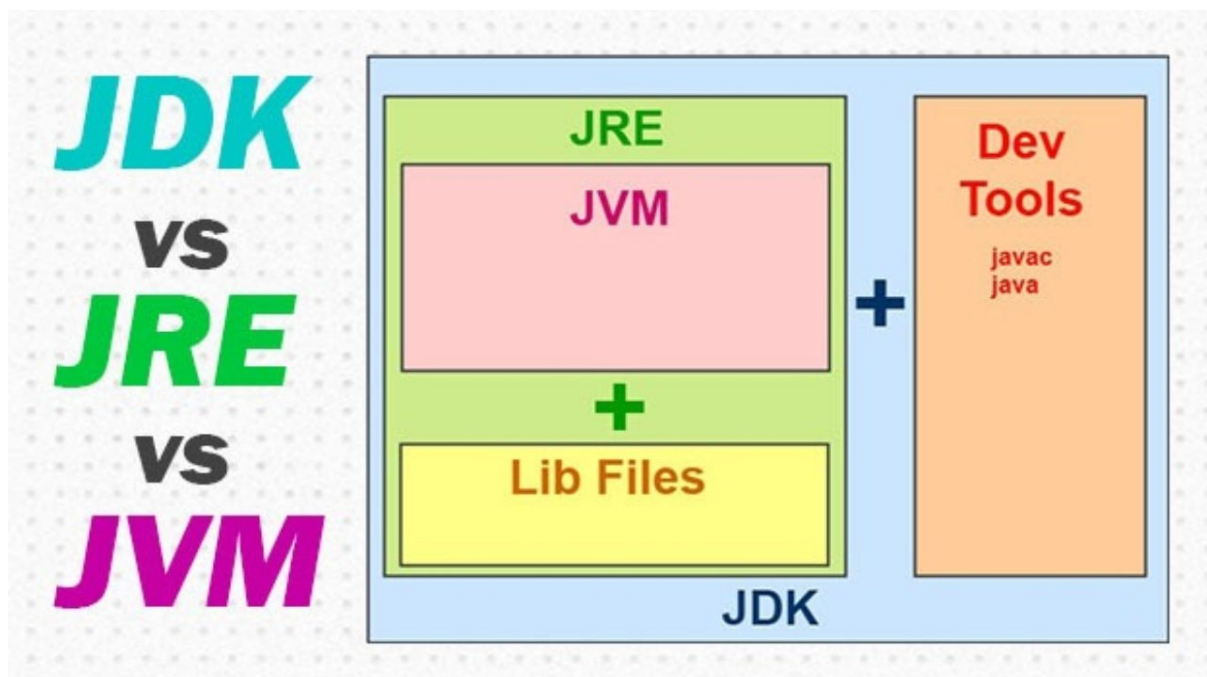
Java is the most popular, widely used object-oriented programming language. By using java, we can develop a variety of applications such as web applications, mobile applications, desktop applications, network applications, and many more. Java is used for developing different applications. Some of them are listed below:

- 1. Banking:** In banking, the Java programming language is used to develop transaction management.

2. **Mobile App Development:** Java can be used to develop mobile applications. Most of the android applications are built using java. So, if you are familiar with java, it will become much easier to develop mobile applications.
3. **Desktop Applications:** We can also create GUI applications in java. Java provides AWT, Swings, and JavaFX to develop desktop applications.
4. **Big Data:** In Big Data, the Hadoop MapReduce framework is written using java.
5. **Web applications:** We can also create web applications using java. The most popular frameworks like Spring, Spring Boot, Hibernate used for developing web applications are based on java.

JDK, JRE, and JVM

JDK, JRE, and JVM are the most important parts of the Java programming language. Without these, you can not develop and run java programs on your machine.



1. **JDK:** JDK stands for Java Development Kit. JDK provides an environment to develop and execute the java program. JDK is a kit that includes two things - Development Tools to provide an environment to develop your java programs and JRE to execute your Java programs.
2. **JRE:** JRE stands for Java Runtime Environment. JRE provides an environment to only run (not develop) the java programs onto your machine. JRE is only used by the end-users of the system. JRE consists of libraries and other files that JVM uses at runtime.
3. **JVM:** JVM stands for Java Virtual Machine, which is a very important part of both JDK and JRE because it is inbuilt in both. Whatever java program you run using JDK and JRE goes into the JVM and JVM is responsible for executing the java program line by line.

main() Method

Before explaining the java main() method, let's first create a simple program to print Hello World. After that, we will explain why the main() method in java is

```
public static void main(String args[]).  
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello World");  
    }  
}
```

- **public:** the public is an access modifier that can be used to specify who can access this main() method. It simply defines the visibility of the method. The JVM calls the main() method outside the class. Therefore it is necessary to make the java main() method as public.
- **static:** static is a keyword in java. We can make static variables and static methods in java with the help of the static keyword. The main advantage of a static method is that we can call this method without creating an instance of

the class. JVM calls the main() method without creating an instance of the class, therefore it is necessary to make the java main() method static.

- **void: void is a return type of method. The java main() method doesn't return any value. Therefore, it is necessary to have a void return type.**
- **main: main is the name of the method. It is a method where program execution starts.**
- **String args[]: String in java is a class that is used to work on Strings and args is a reference variable that refers to an array of type String. If you want to pass the argument through the command line then it is necessary to make the argument of the main() method as String args[].**

Comments in Java

Comments are statements that are not executed by the compiler. Comments make the program more human-readable by including the details of the code involved. The proper use of comments makes maintenance and debugging of the code easier.

Types of Comments

There are three types of comments in java

- Single line comments
- Multi-line comments
- Documentation comments
- **Single line comment:** The single line comment is used to comment only one line. A beginner-level programmer uses mostly single-line comments for describing the code functionality.

Syntax:

// write your comment here

Example:

// Java program to show single line comment

```
public class HelloWorld {  
    public static void main(String args[]) {  
  
        // Print "Hello World" on console  
        System.out.println("Hello World");  
    }  
}
```

- **Multi-line comments:** The multi-line comments are used to comment multiple lines of code.

Syntax:

*/**

This

is

multiline

comment

**/*

Example:

```
public class HelloWorld {  
    public static void main(String args[]) {  
  
        /*  
  
        Here, we have  
        declared a  
        variable and we are  
        printing its value  
  
        */  
        int a = 10;  
        System.out.println(a);  
    }  
}
```

- **Documentation Comments:** This type of comment is used generally when we write the code for projects. It helps to generate a documentation page for reference, which can be used for getting information about methods present, their parameters, etc.

Syntax:

```
/**Comment start  
*  
*tags are used in order to specify a parameter
```

**or method or heading.*

**HTML tags can also be used*

**such as <h1>*

**comment ends*/*

Example:

// Java program to illustrate documentation comments

*/***

** Find product of four numbers! The FindPro program implements an application*

** that simply calculates product of four integers and Prints the output on the*

** screen.*

** @author Prashant Srivastava*

** @version 1.0*

** @since 2021-02-26*

**/*

public class FindPro {

*/***

** This method is used to find the product of four integers.*

** @param num1 This is the first parameter to FindPro method*

** @param num2 This is the second parameter to FindPro method*

** @param num3 This is the third parameter to FindPro method*

** @param num4 This is the fourth parameter to FindPro method*

** @return int This returns average of numA, numB and numC.*

**/*

```

public int FindPro(int num1, int num2, int num3, int num4) {
    return (num1 * num2 * num3 * num4);
}

/**
 * This is the main method which makes use of the FindPro method.
 *
 * @param args Unused.
 * @return Nothing.
 */

public static void main(String args[]) {
    FindPro obj = new FindPro();
    int pro = obj.FindPro(10, 20, 30, 40);

    System.out.println("Product of 10, 20, 30 and 40 is : " + pro);
}
}

```

Output:

Product of 10, 20, 30 and 40 is : 240000

Introduction and Declaration of Variables

Introduction to Variables

A **Variable** is a name given to a memory location. It is used to store a value that may vary. Java is a statically typed language, and hence, all the variables are declared before use.

Variable Declaration

In Java, we can declare variables as follows:

- **type:** Type of the data that can be stored in this variable. It can be int, float, double, etc.
- **name:** Name given to the variable.

`data_type variable_name;`

Example: `int x;`

In this way, we can only create a variable in the memory location. Currently, it doesn't have any value. We can assign the value in this variable by using two ways:

- By using variable initialization.
- By taking input

Here, we have discussed only the first way, i.e. variable initialization. We will discuss the second way later.

`data_type variable_name = value;`

Example: `int x = 10;`

Variables naming Convention in Java

- A variable name should be short and meaningful.
- It should begin with a lowercase letter.

- It can begin with special characters such as _ (underscore) and \$ (dollar) sign.
- If the variable name contains multiple words, then use the camel case, i.e. variable name should start with a lowercase letter followed by an uppercase letter. For eg: codingNinja, camelCase.
- Always try to avoid single character variable names such as i, j, and k except for the temporary variables.
- A variable name can not contain whitespaces.
- We can't use keywords(pre-defined literals) as the variable names.

Java Keywords: Keywords in Java are also known as reserved words. These are the predefined words therefore they can't be used as a variable name. If we will use keywords as a variable name, the result will be a compile-time error. The list of all the Java Keywords is given below

.

abstract	final	protected
assert	finally	public
boolean	for	return
break	float	short
byte	if	static
catch	implements	super
char	import	switch
case	instanceOf	synchronised
class	int	this
continue	interface	throw
default	long	throws
do	native	transient
double	new	try
else	null	void
enum	package	volatile
extends	private	while

Examples:

Correct

```
int x = 10;
```

```
int _x = 10;
```

```
int $x = 10;
```

```
Int x1 = 10;
```

Incorrect

```
int 1x = 10;
```

```
int num ber = 10;
```

Data types in Java

The data type defines the type of value that can be stored in a variable. For example, if a variable has an ***int*** data type, it can only store an integer value. In java, there are two categories of data types.

1. **Primitive Data Type:** A primitive data type is predefined by the language and is named by a keyword or reserved keyword. There are eight types of primitive data types in java such as boolean, char, int, short, byte, long, float, and double.
 - **boolean:** boolean data type specifies only one bit of information and it is used to store only two possible values either true or false.
 - **byte:** byte data type is 8 bit signed two's complement integer. Its value lies between -128 to 127. It has a minimum value of -128 and a maximum value of 127 (inclusive). The byte data type is most commonly used to save memory in large arrays.
 - **short:** short data type is a 16-bit signed two's complement integer. It can hold any number between -32768 to 32767 (inclusive). Like byte data type, it is commonly used to save memory in large arrays.
 - **int:** int data type is 32-bit signed two's complement integer. It can hold the number between -2,147,483,648 to 2,147,483,648. The default value of the int data type is 0.
 - **long:** long data type is 64-bit two's complement integer. It can hold the number between -2^{63} to $2^{63}-1$. The default value of long data type is 0.

- **float:** float data type is used to store floating-point numbers. The float data type is a single-precision 32-bit IEEE 754 floating-point. It can hold 6 to 7 decimal digits. It is recommended to use float instead of double if you need to save memory in large arrays of floating-point numbers. The default value of float is 0.0f.
- **double:** double data type is generally used to store decimal values. The double data type is a double-precision 64-bit IEEE 754 floating-point. For decimal values, this data type is generally the default choice. The default value of double is 0.0d.
- **char:** The char data type is used to store characters. The char data type is a single 16-bit Unicode character.

2. Non-Primitive Data Type: Non-Primitive data type refers to the objects. ArrayList and String are some of the examples of Non-Primitive data type. We will discuss the Non-Primitive data type later.

Example

// Primitive Data Types

int price = 5000; // Integer Value

float rateOfInterest = 5.99f; // Floating point number

char ch = 'a'; // Character

// Non-Primitive Data Types

String str = "Coding Ninjas"; // String

Scope of Variables in Java

The variable scope is the part of the program where the variable is accessible. The scope of the variable can be determined at compile time. There are mainly two types of variable scope.

1) Local Variables Scope: A variable that is defined inside a block, method body, or constructor is called a local variable. These variables can't be accessed outside the method.

Example:

```
public class VariableScope {  
    void method() {  
        // local variable (Method Level Scope)  
        // This method can't be accessed outside  
        // method body.  
        int x;  
    }  
}
```

2) Member/Class Level Variable Scope: A variable that is declared inside the class but outside the method body, block, or constructor is known as member/class level variable. These variables can be directly accessed anywhere in the class.

Example:

```
class VariableScope {  
    // variable defined inside the class  
    int x;  
}  
  
public class VariableScopeDemo {
```

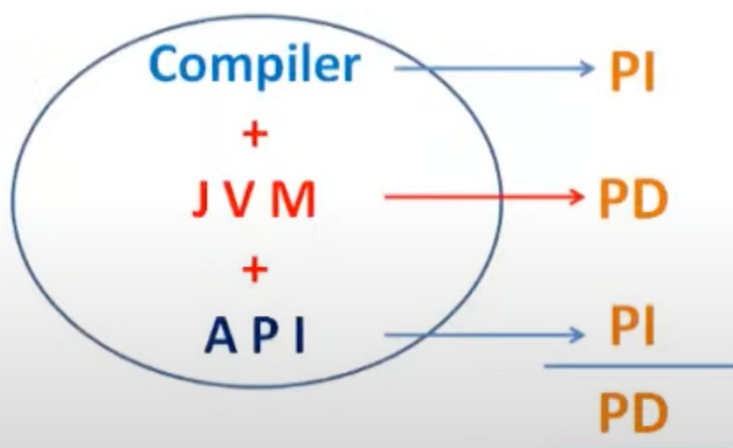
```
public static void main(String args[]) {  
  
    // Creating VariableScope class object  
    VariableScope obj = new VariableScope();  
  
    // Assigning values in the variable  
    obj.x = 10;  
  
    // Printing the value  
    System.out.println(obj.x);  
}  
}
```

Output:

10

Important facts on Java program and Java software

1. Java Slogan: WORA- Write Once Run Anywhere
2. Compiler is responsible to take Java source code, verifies syntaxes, if no errors convert source into bytecode and saving this bytecode in a separate new file with ".class" extension with class name as file name.
3. JVM is responsible to take Java bytecode, execute this bytecode by converting it into current OS ML. After execution generated ML is destroyed.
4. Java supports only 2 extension files .java and .class.
5. In Java we do not have a file with machine language code means Java does not support .exe file, because it is platform dependent, it is not safe, easily effected by virus, and more over always compiled code is downloaded from server, hence ML code saved in client system is useless.
6. Java compiler is OS independent because both input and output files source code and bytecode are Java related files.
7. JVM is OS dependent, because it should generate OS dependent ML from the given bytecode.
8. Java software contains



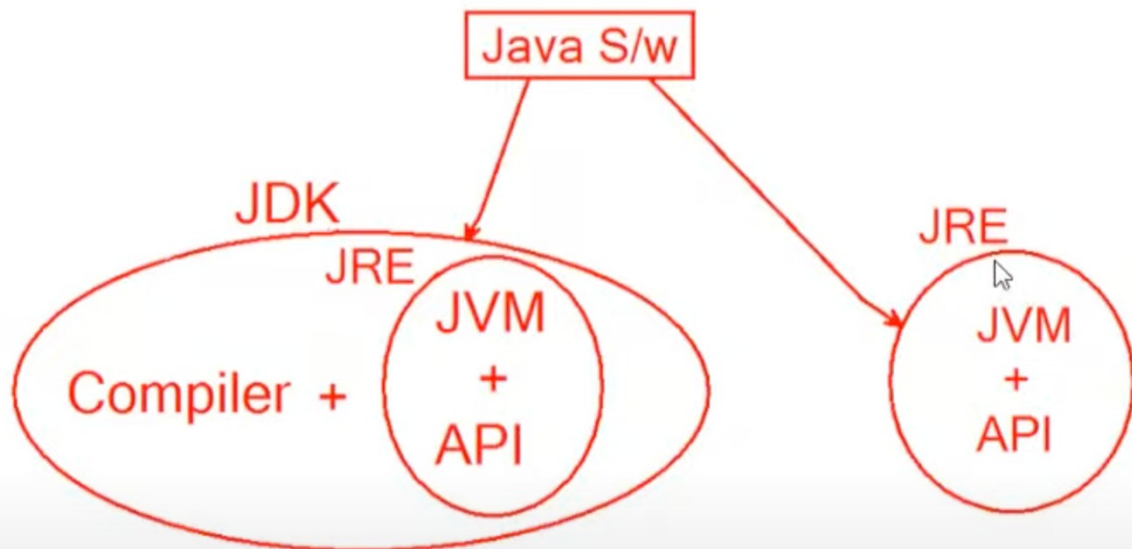
9. Java software is platform dependent because JVM is platform dependent. Because Java Software is platform dependent, we will have separate Java software for every OS. Hence based on our system OS and processor we must download and install Java software
10. Java program is platform independent and Java software is platform dependent, both because of JVM.
 - java program platform independent, because JVM is available for every OS, so Java program can be executed in all OS
 - Java software is platform dependent, because we have separate JVM for every OS, so the same Java software can not be installed on all OS.
11. For all OS, JVM is developed by SUN(Oracle), but not by OS Vendors. So, JVM will not be installed inbuilt along with OS installation. We must install JVM explicitly in our computer after OS installation.

12. Programmer is responsible for developing, compiling and executing a Java program, Whereas

13. Customer /Client is responsible for only executing Java program.

14. Based on programmer and customer responsibilities Java software is divided into two types

Type of Java Softwares



Why public JRE is removed in Java 11v?

- In modern web applications development the client UI pages are developed by using HTML, CSS and JavaScript or React, for running HTML pages JRE is not required just Browser is sufficient
- Hence JRE is not being used in a client system so it is removed in Java 11v